FIS-by-Step: Visualization of the Fast Index Scan for Nearest Neighbor Queries

Elke Achtert, Dominik Schwald

Institute for Computer Science, University of Munich, Germany
{achtert,schwald}@dbs.ifi.lmu.de

Abstract. Many different index structures have been proposed for spatial databases to support efficient query processing. However, most of these index structures suffer from an exponential dependency in processing time upon the dimensionality of the data objects. Due to this fact, an alternative approach for query processing on high-dimensional data is simply to perform a sequential scan over the entire data set. This approach often yields in lower I/O costs than using a multi-dimensional index. The Fast Index Scan combines these two techniques and optimizes the number and order of blocks which are processed in a single chained I/O operation. In this demonstration we present a tool called FIS-by-Step which visualizes the single I/O operations during a Fast Index Scan while processing a nearest neighbor query. FIS-by-Step assists the development and evaluation of new cost models for the Fast Index Scan by providing user significant information about the applied page access strategy in each step of the algorithm.

1 Introduction

A large number of index structures for high-dimensional data have been proposed in previous years, cf. [2] for details. However, for sufficiently high dimensional data the complexity of similarity queries on multidimensional index structures is still far away from being logarithmic. Moreover, simple query processing techniques based on a sequential scan of the data are often able to outperform approaches based on sophisticated index structures. This is due to fact that usual index structures access data in too small portions and therefore cause lots of I/O accesses. The Fast Index Scan proposed in [1] subsumes the advantages of indexes and scan based methods in an optimal way. The algorithm collects accesses to neighboring pages and performs chained I/O requests, where the length of the chains are determined according to a cost model. The benefit of this chained I/O processing is that the seek costs–the main part of the total I/O costs–have to be paid only once. The authors have shown that the Fast Index Scan clearly outperforms both, the sequential scan as well as the Hjaltason and Samet algorithm which is typically used for processing nearest neighbor queries.

In this demonstration we present a tool called FIS-by-Step to visualize the single chained I/O operations during a nearest neighbor query by applying the Fast Index Scan on top of an R-Tree. FIS-by-Step displays the applied page

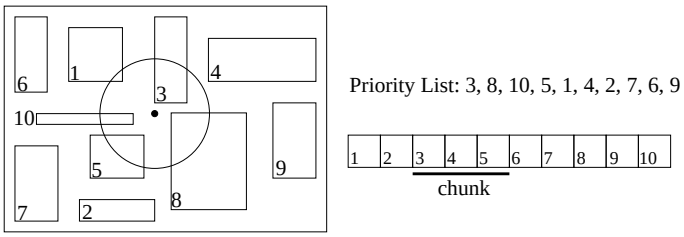

Fig. 1. The Fast Index Scan for nearest neighbor queries

access strategy in each step of the algorithm and provides user significant statistical information. The step-by-step visualization is very useful in a lot of cases, e.g. for teaching and explaining the function of the Fast Index Scan, for visual evaluation of the applied strategies or for development of new strategies.

The remainder of this paper is organized as follows: The concepts of the Fast Index Scan are described in Sect. 2. In Sect. 3 we demonstrate our tool FIS-by-Step for visualizing the I/O operations during a Fast Index Scan.

2 The Fast Index Scan

As the Fast Index Scan has been evaluated in [1] on top of the IQ-Tree, it can be applied to any R-Tree like spatial index structure that consists of only one directory level. Usually nearest neighbor queries are evaluated by the algorithm of Hjaltason and Samet (HS) [3], which has been proven to be optimal w.r.t. the number of accessed pages. Unlike the original HS algorithm which loads and processes one page after the other, the Fast Index Scan adapts the HS algorithm and tries to chain I/O operations for subsequent pages on disk and optimizes the number and order of pages which are processed in a single I/O-operation.

The HS algorithm keeps a priority list of all data pages in increasing order to their distance to the query point. For all pages p_i in the priority queue there exists a certain probability that p_i has to be loaded to answer the query. The idea of the Fast Index Scan is to load in each step a chunk of neighboring pages with a sufficient high probability instead of loading only one page, as the HS algorithm would do. In [1] the authors proposed a stochastic model to estimate the probability of a page to be accessed during a nearest neighbor query. Based on this access probability the cost balance of a page can be determined. A negative cost balance indicates that it is likely to be "profitable" to load the page in the current chunk additionally. This is given if the additional transfer costs to read the page in the current chunk are less than the estimated costs to read the page later in the algorithm. In Fig. 1 the page strategy of the Fast Index Scan is visualized: starting with page 3 the Fast Index Scan extends the chunk and reads page 4 and 5 additionally, because page 5 has a very high probability to be necessary to answer the query. Thus, reading page 4 and 5 in the current

(a) Step 1

(b) Step 2

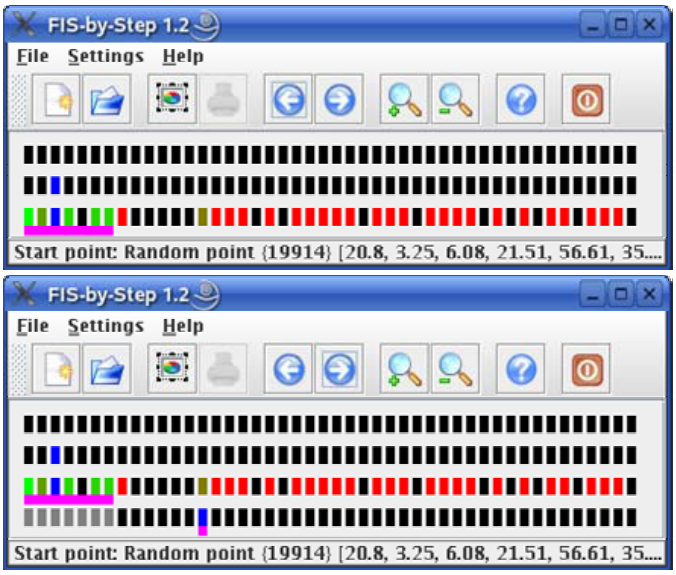

Fig. 2. Screenshots of the FIS-by-Step application

chunk is less expensive than loading page 5 in all probability later and causing additional seek costs.

3 Visualization of the Fast Index Scan

The main purpose of the FIS-by-Step application is to show step-by-step how the Fast Index Scan solves a nearest neighbor query. Figure 2 shows screenshots of our application to explain how FIS-by-Step works.

Before running a query, it is possible to change some settings like the pagesize in order to adjust the application to the data. After choosing a data file (a CSV file of hyperpoints), the first line of black rectangles appears in the main window of the application. Each of these rectangles represents a page on the disk, where the order of the rectangles is identical with the order of the pages on disk. After selecting the query point (which can be the first point of the data, a random point of the data, or any given point) the second line of rectangles appears, again showing all pages, but now there is one blue rectangle: This is the page that is the nearest one to the query point.

The third line appears after using the "Next Step" button. This is the first step of the Fast Index Scan: The access probability is calculated for all pages. The different colors of the rectangles indicate the access probability of the pages: Black indicates an access probability of 0%, i.e. these pages need not to be read. Grey pages have been already processed and thus also have an access probability of 0%. A blue page is the nearest unprocessed page to the query point and therefore has an access probability of 100%. All other pages (with red to green

Fig. 3. Statistics about the solved query

color) have access probabilities between 0% and 100% and might have to be read during the algorithm. As illustrated in Fig. 2(a), in this step of our example 7 pages (underlined magenta) are read by the Fast Index Scan.

After using the "Next Step" button again, two things can happen: Either the query is solved and some statistics are displayed, or the query is not solved yet, so it is necessary to read some more pages as shown in Fig. 2(b). Note that all pages that have been read in the last step are now colored gray, as their access probability is now 0%. A lot of red colored pages from the first step are now black, since they have a larger distance to the query point than the nearest point of the already processed pages. Also there is a new blue page, i.e. a page with an access probability of 100%. This page is the one that is the nearest one to the query point, as all already processed pages are ignored. After this step the example query is solved, thus after using the "Next Step" button there does not appear a new line of rectangles, but a popup window, showing some statistics about the query (cf. Fig. 3). The statistical information consists of the number of accessed pages and the I/O time for solving the query using the Fast Index Scan in comparison to use the HS algorithm or the sequential scan of the data set, respectively. As the statistic shows, the Fast Index Scan outperforms the HS algorithm as well as the sequential scan.

As mentioned above, the primary objective of our FIS-by-Step application is the step-by-step visualization of the Fast Index Scan. This stepwise visualization is very useful in a lot of cases, e.g. for teaching and explaining the Fast Index Scan. FIS-by-Step supports the visual evaluation, comparison and improvement of strategies for building chunks for chained I/O operations, layout of pages on disk, and ordering pages for processing in CPU. Furthermore, FIS-by-Step assists the development of new strategies, as the advantages and disadvantages of a strategy for a given data are shown directly.

References

1. S. Berchtold, C. Böhm, H. V. Jagadish, H.-P. Kriegel, and J. Sander. Independent Quantization: An index compression technique for high-dimensional data spaces. In *Proc. ICDE*, 2000.
2. C. Böhm, S. Berchtold, and D. A. Keim. Searching in high-dimensional spaces: Index structures for improving the performance of multimedia databases. *ACM Computing Surveys*, 33(3), 2001.
3. G. R. Hjaltason and H. Samet. Ranking in spatial databases. In *Proc. SSD*, 1995.